地铁列车环境中多依赖复杂事件处理研究 *

乔雅正 [a]，程良伦 [a]，王 涛 [b]，黄田安 [a]

(广东工业大学 a. 计算机学院; b. 自动化学院, 广州 510006)

摘 要：针对多源海量实时数据的复杂事件检测中，原始事件流的分流处理存在事件检测准确率低及效率慢的问题，提出一种基于事件树的复杂事件检测方法。首先给出事件依赖关系的明确定义，然后根据原子事件间存在的多依赖关系生成原子事件树，以事件树为节点构造依赖事件树链表，提升复杂事件处理引擎的有效检测次数，使得事件检测的匹配效率得到提升。同时该方法减少了事件检测过程的内存消耗，提高了事件检测的吞吐量。仿真实验与案例研究证明了提出方法在海量数据处理上的优异性及可行性。

关键词：复杂事件处理；依赖事件分流；事件树；地铁列车

中图分类号：TP311　　doi: 10.3969/j.issn.1001-3695.2018.02.0087

Study on multi-dependency complex event processing in subway train environment

Qiao Yazheng[a], Cheng Lianglun[b], Wang Tao[b], Huang Tian'an[a]

(*a. Dept. College of Computer Science & Technology Guangdong University of Technology* 510006, *China*; *b. Dept. College of Automation Guangdong University of Technology* 510006, *China*;)

Abstract: For complex event detection in the mass real-time data from multiple sources, there is a problem of low accuracy and inefficiency in the triage of original event streams. This paper proposed a method of complex event detection based on an event tree. Firstly, defining dependencies between events, then take full account of the multi-dependencies between atomic events to generate atomic event trees and form the list with the dependent event trees, increasing the number of effective detection of complex event processing engines, such that the matching efficiency of event detection is improved. Meanwhile, this method reduces the memory consumption and improves the throughput of event detection. Simulation experiments and case studies demonstrate the advantages and feasibility of this method on massive data processing.

Key words: complex event processing; triage of dependent events; event tree; subway train

0 引言

在地铁列车的运行环境中，大量感知设备用于监控列车的运行过程中的环境因素、设备性能、状态变化等等，从而产生了海量关系复杂的原始数据流。事件的处理对于从原始数据流中发现知识是至关重要的[1]，复杂事件处理（complex event processing， CEP）[2]作为一种新兴的基于事件流的技术，通常在特定的上下文环境下，从异构事件源的原子事件中检测出存在业务价值的复杂事件模式，进而实现自动化的解决方案或警报机制等。

CEP 本质是从匹配识别事件的规则中选取并执行规则，所以现有的 CEP 处理大多是将多个异构数据源的事件流合并后顺序地执行匹配规则，比如下列四种经典的 CEP 检测方法：基于自动机复杂事件检测方法[3]，基于 Petri 网复杂事件检测方法[4]，基于匹配树复杂事件检测方法[5]和基于有向图复杂事件检测方法[6]。但在面临地铁列车产生的实时多源且存在依赖关系的原子事件流时，CEP 引擎以集中串行顺序的处理方式是不能满足上层应用的需求的。在这种情形下，越来越多的研究开始关注于复杂事件的分流处理问题。

如 Schultz-Moller 等人[7]提出一种在集群上的多个节点上分配查询处理的方法，在吞吐量方面提高了系统的性能，但是该方法是通过将多源事件传输到中央节点后路由到处理节点，方法的优化只在中央节点上进行，处理节点的检测效率较低，同时路由的通信成本较高。Akdere 等人[8]提出一种复杂事件检

收稿日期：2018-02-21；**修回日期：**2018-03-22　**基金项目：**国家自然科学基金青年科学基金资助项目（61502110）；粤港共性技术招标项目（2013B010134011）；广东省科技计划资助项目（2016B090918045）；广东省科技计划资助项目（2017B090901019）

作者简介：乔雅正（1990-），男，河南南阳人，硕士研究生，主要研究方向为复杂事件检测，物联网与物理信息融合系统（363793059@qq.com）；程良伦（1965-），男，湖北黄石人，教授，博导，主要研究方向为网络控制与系统集成、网络与信息化控制、物联网与物理信息融合系统；王涛（1983-），男，广东潮州人，副教授，主要研究方向为传感网与物联网、资助物联、物理信息融合系统；黄田安（1992-），男，安徽芜湖人，硕士研究生，主要研究方向为大数据、数据挖掘。

测的协调器来拉取和推送事件进行分析，通过对 FSM 状态重新排序或重写来标记分发，有效降低通信成本与检测延迟，但是该方法的时间复杂度是指数级，仅适用于小规模事件检测。Chen 等人[9]设计了 14 种复杂事件处理操作符，给出了基于操作符的事件负载分流方法，相比于集中串行顺序的检测速度有很大程度的提升，但在面对地铁列车的环境，不考虑事件间的依赖关系，该方法的检测效率不足以满足需求。针对以上问题，本文将研究一种高效率，高吞吐，低内存的复杂事件检测方法。

针对地铁列车环境中海量多依赖事件流的高效实时检测技术是保证列车正常安全运行的关键技术。因此，本文提出基于事件树的事件负载分流方法及引擎架构，定义异构事件源间的多依赖关系及地铁环境中的事件分类，以原子事件构成的事件树为节点构造存在多依赖关系的事件树链表，减少事件检测过程中的冗余检测，从而提高复杂事件检测效率。仿真实验结果证明了该方法的高效性。

1 多依赖事件

1.1 事件的依赖关系

事件作为复杂事件检测的对象，是复杂事件检测的基本组成部分[10]。为了实现由简单事件生成高级事件，只有更好的描述事件间的关系才能实现复杂事件检测。基于此本文将事件间的依赖关系分为两类：分类关系和非分类关系，并给出事件间的关系定义如下：

定义 1　分类关系。假如有事件类 EC_1 和 EC_2，且 EC_1 和 EC_2 间有分类关系，如果 EC_2⊂EC_1，我们称 EC1 为上位事件类，EC2 为下位事件类。

定义 2　非分类关系。事件的非分类关系包括以下四种主要类型：

a）复合关系。如果复合事件 e 可以分解为多个子事件 e_i（i> 0），当这些子事件完成时，表示复合事件 e 完成，则复合事件 e 和子事件 e_i（i> 0）之间存在复合关系。如果事件类 EC1 的每个事件实例都是由事件类 EC2 的实例和其他事件类的事件实例构成，即 EC1 由 EC2 组成，记为 EC_2<EC_1。

b）因果关系。如果事件类 EC1 的事件实例发生，事件类 EC2 的事件实例在指定的概率阈值以上有可能发生，则事件类 EC1 与事件类 EC2 间存在因果关系。即 EC1 是原因事件类，EC2 是结果事件类，记为 EC_1⟹EC_2。

c）连接关系。如果在一定的时间内，事件类 EC1 和 EC2 的事件实例按照先后时间顺序产生，EC2 作为 EC1 逻辑运行的结果，则事件类 EC1 与 EC2 存在连接关系。即 EC1 连接 EC2，记为 EC_2▹EC_1。

d）伴随关系。如果在一定的时间内，事件类 EC1 与事件类 EC2 同时发生，且发生的概率高于某规定的概率值，则事件类 EC1 与 EC2 间存在伴随关系。即 EC1 伴随于 EC2，记为 EC_2 //EC_1。

其中分类关系，反映的事件间的静态特征关系，指异构事件源间的不同类别之间的分类关系，如 RFID 读写器、无线传感器、网络监控设备、活动数据库等产生的原始数据等。非分类关系反映的是事件在时间顺序关系上的深层次的语义关系，如关联关系，分类关系，聚类关系等。

1.2 地铁运行环境中的事件分类

目前，以广州地铁为例，其运营总长度有 400KM，在复杂多变的地铁运行环境中为了保持地铁列车的持久安全运行，需要对列车运行的环境实时掌控，面向列车运行环境中的海量数据多源动态数据流复杂事件高效检测机制实现了对数据的实时智能处理[11]。

在列车的运行环境诸多因素中，本文将环境中的感知对象分为固定对象、移动对象、环境对象。根据感知对象的特征，我们将其分解为微观、中观和宏观三个层面，如表 1 所示。

表 1　地铁环境中的对象分类表

对象形态	对象特征		
	微观	中观	宏观
环境对象	传感器上电压、电流等数据的变化	温度、湿度、粉尘等单个参数的变化	环境影响，地理影响等综合因素
移动对象		列车、人群等单个参数的变化	列车运行，乘客行为等综合因素
固定对象		位移、形变等单个参数的变化	基础设施等综合因素

其中固定对象指其位置不随时间的变化而变化的感知对象，包括隧道，轨道及站台建筑等；移动对象指随着时间变化而不断变化的感知对象，包括机车，乘客等。不同的感知对象间产生的感知事件虽然由不同对象产生，但参照定义 1 和 2 将其之间的关联性分为：

a）空间关联性。运行环境中的参与个体具有相对的空间变化。如列车在高速通过某空间时，其运行速度直接影响路基的沉降大小。

b）时间关联性。运行环境中的参与个体随时间的变化而产生变化。如在固定上下班高峰时段，站台的人流量是会有增加。

c）不确定关联性。运行的环境会因诸多不确定的因素产生变化。如突发事件，随机干扰和主观影响等不确定的因素造成的影响。

2 基于事件树的事件划分方法

前文已经说明，面对多源海量的实时事件流，传统的 CEP 以集中处理方式无法及时处理，不考虑事件间关系的分流方法同样效率较低。所以根据事件间的依赖关系将事件流划分到不同 CEP 引擎处理，是提升复杂事件检测效率的有效方法。如何将海量实时的原始事件流快速、高效地划分为多条内含多依赖关系的事件流是实现基于事件树的事件划分方法的关键。

利用树结构的深度，高度，子树及根叶节点的变化可以更

好的处理原始事件流动态实时感知的特性，如 Left Deep Tree[12]，Tree-NFA[13]，CPI-Tree[14]等。因此我们以事件树的结构为基础进行事件分流，依照上文中提到的事件间依赖关系，定义事件之间不同类别的分类关系及深层的语义关系，同时虑事件优先级和层次结构，提出一种解决事件间依赖性的复杂事件分流方法。

2.1 算法思想及实现

1）算法思想　在事件的分流过程中，考虑事件间的依赖性，将存在相互依赖的事件分流到同一个 CEP 引擎中处理，相比于不考虑事件依赖关系，按照集中顺序固定分割的分流方法，CEP 引擎的检测效率会有提升。在分流的过程中考虑不同事件种类的优先级，优先级高的事件对于得到复合事件有较高的重要性。

2）算法步骤

a)根据用户需求的对应业务进行优先级划分，其中要求处理时间短的业务对应的优先级越高。

b)参照 a)中定义的优先级，在事件的有效长度窗口内遍历构造事件树。

（a）以不同种类的事件为叶子节点，并为每个叶子节点标注其事件的对应优先级值；先给出一个空的根节点。

（b）循环比较（a）中构造的每个事件树结构叶子节点的最高优先级值，并将最大的优先级值赋值给当前树的根节点。如图 1 所示。

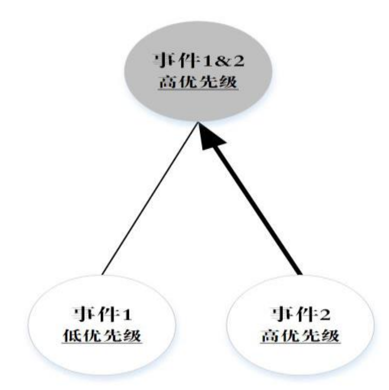


图 1　事件树结构

（c）在事件的有效长度窗口内按照上述方式生成所有的事件树。

c)遍历 b)中生成的事件树，记录每个子树根节点的优先级值，按照定义 1、2 记录节点事件间的依赖关系，将存在依赖关系的事件树存放到一个待处理队列中，等待 CEP 引擎的请求处理。

3）算法实现

在上述的算法步骤中，依赖事件树的生成较为重要，是实现事件分流的基础，其中事件树生成算法伪代码如算法 1 所示。

Algorithm1. Event Tree Generation Algorithm

Input: 1 sequence of events collected by the sensor (include n events);
2 defines the size of the event tree (here scale)

```
new Tree tree = {};
define priority for collecting events according application requirements
for i = 1; i <= n; i ++
    put the ith event as a leaf node to the tree
    if the priority of the ith event > root node priority
        the root node priority value = the priority value of the ith event
    endif
    else
        if the number of leaf nodes = scale;
            out the current structure of the completed event tree
            new Tree tree = {};
    end
end
```

Output. the generated event tree

其中 scale 的值取决于 CEP 引擎的处理性能。根据输入的事件序列，循环构造多个事件树，我们使用 hashmap 的方式实现事件树的存储，可以节省海量数据的插入、查找时间。这些生成的事件树之间是存在依赖关系的，因此需要将有依赖关系的事件树推送到一个 CEP 引擎中，所以将依赖的事件树存放在一个事件树链表中，依赖事件树链表生成算法伪代码如算法 2 所示。

Algorithm2. Event Tree Generation Algorithm

Input: m event trees generated by the sequence of events

```
new List list = {};
for i = 1; i <= m; i ++
    for j = i+1; j<=m; j++;
        if the ith tree has the same children as the jth tree
            put the jth tree in the same list as the ith tree;
            remove the jth tree;
        endif
        else
            new List list = {};
            put the jth in the new list;
        end
end
```

Output. the generated event tree list

算法 2 中虽然存在双重的 for 循环，但是在判断过程中，如果存在一个相同的原子事件节点，我们就将整棵事件树添加到 list 中，然后从集合中移除该事件树，所以这个循环过程是对比量不断减少的过程，对事件的分流效率影响不大。

2.2 事件分流引擎架构

本文采用的事件分流引擎架构如图 2 所示。在本文所研究的地铁列车环境中，其产生的事件是多种多样的，包括传感器或 RFID 产生的数据流，GPS 位置信息，天气信息等等。我们把这些多种类的事件合在一起成为事件云。该架构主要包括事件

云，事件流划分器，CEP 引擎集群等，其中主要事件划分器部分是本文算法的主要体现。

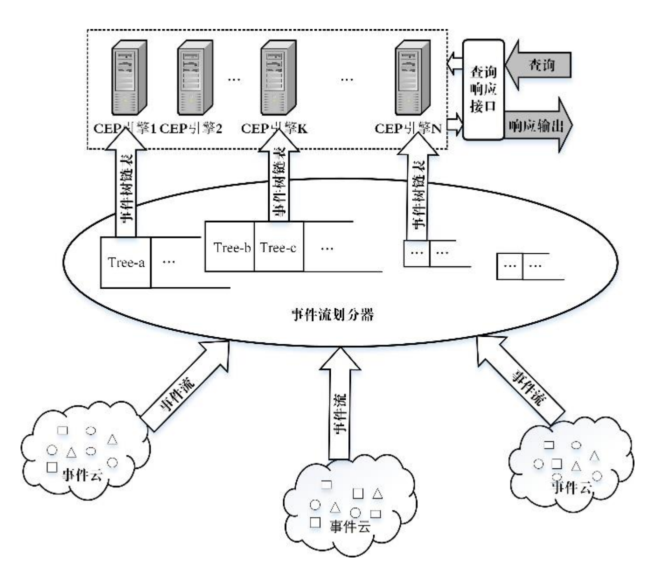


图 2 事件分流引擎架构

事件划分器负责对事件云中的实时数据流进行依赖关系的分流处理。在图 2 中，当查询操作发生，根据算法 1 从原始事件云中构造事件树，遍历事件树中的叶子节点事件，若两棵事件树中包含如定义 1，2 的依赖关系，则按照算法 2 将两棵树添加到同一个事件树链表中，如图 2 中的 Tree-b 与 Tree-c；否则将该事件树添加到新的事件树链表，如图 2 中 Tree-a。当该滑动窗口内的事件构造完毕后，将这些事件树链表分配给不同的 CEP 引擎处理，最终输出查询结果。

3 实验分析与案例研究

本文选择以 Java 开发的复杂事件处理引擎 Esper，Esper 可以结合多种数据源的数据对信息流进行监测、分析从推理出一些复杂的事件或模式[15]。数据源以广州地铁 2015 年 3 号线夏滘车辆段的运行时无线传感网检测数据为事件源，事件输入流类型包括温度事件流，湿度事件流，光照事件流，粉尘事件流，电压事件流等 6 种。

3.1 实验分析

实验的环境为在实验室条件下搭建的 3 个节点的集群环境，每个节点的操作系统为 Centos7.2，Java 版本为 JDK 1.8.0_91，每个节点配置为CPU Intel Xeon E5-2650 2.6 GHz，内存 20 GB，在每个节点上搭建 Esper 处理引擎为 6.1.0。本次实验的事件基数为 10000，滑动窗口大小为 100。。

为了验证基于事件树的多依赖事件划分方法的高效性和实时性，实验主要对比 Schultz-Moller 等人的多个节点上分配查询处理的方法[7]、Chen 等人的基于操作符的事件分流方法[9]和本文提出的方法在平均匹配效率、事件处理吞吐量及内存占用空间三个指标上的优势。为方面起见，下文中的算法 1 表示 Schultz-Moller 等人的方法，算法 2 表示 Chen 等人的方法，算法 3 表示本文研究方法。

平均匹配效率指的是处理每个输入事件的时间消耗，时间消耗越小意味着有更高的响应性能。如图 3，在单事件流下，三种算法的响应时间区分不大，但在输入事件流种类数超过 3 时，使用事件树分流的算法 3 有更优的匹配效率；当种类数变为 6 时，算法 3 响应时间明显优于算法 1、2。这是因为算法 3 在进行事件分流时，充分考虑原子事件间存在的多依赖关系，以存在依赖关系原子事件构成事件树为节点构造事件树链表，将不同的依赖事件树链表分配到不同 CEP 引擎，减少不相关事件的冗余检测和计算，因而节省了检测时间。

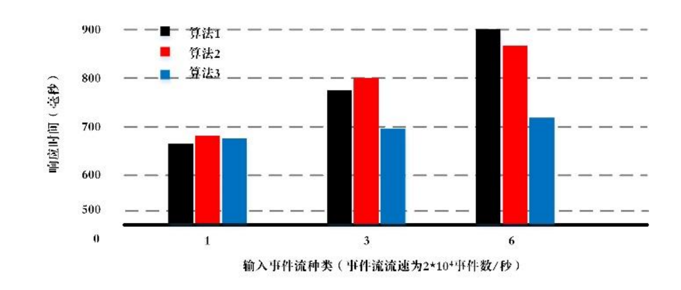


图 3 不同输入事件种类下的算法匹配率对比

事件处理吞吐量反映系统在单位时间内能处理的事件规模，以每秒钟处理的事件数量来量化表示。如图 4，输入采用六种类型的混合事件流，在相同输入规模下，算法 3 相比算法 1、2 有更大的事件吞吐处理能力。在较大的输入流（$8*10^4$）环境下，算法 3 的事件处理吞吐量约比算法 1、2 高出 30%。这是因为算法 3 将原子事件通过事件间的依赖关系进行连接，减少低价值事件检测对系统计算资源造成的消耗，从而提高了系统的处理能力，增加了事件处理的吞吐量。但是随着事件规模不断增大，处理节点之间的通信也消耗了一定资源，因此事件处理吞吐量并不能随着事件规模的增大而无限提升。

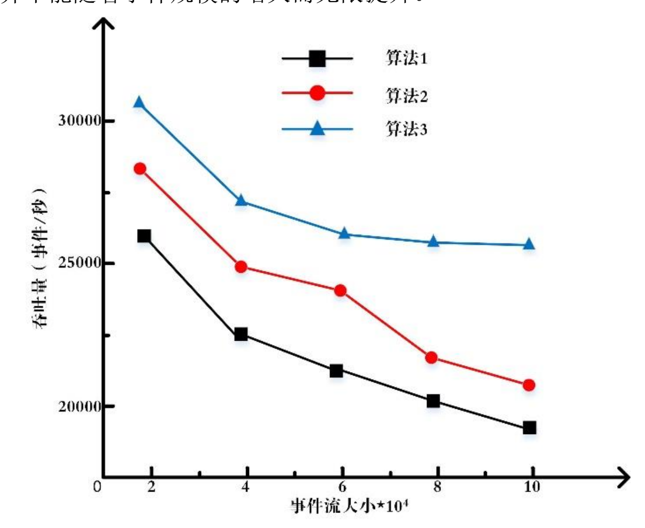


图 4 三种算法吞吐量对比

算法运行时的内存占用空间作为衡量算法性能的重要指标，如图 5 所示，在相同的测试环境及事件规模下，算法 3 所占用的内存空间大小更优于算法 1、2，究其原因是在对原子事件流进行事件树链表的构造过程中，存在依赖关系的事件被加入链表，而少量不存在依赖关系的原子事件被过滤掉，因此省了内存的消耗。

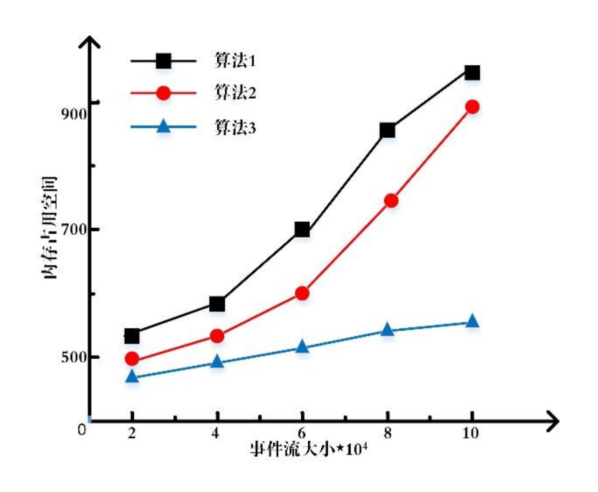


图 5 三种算法内存占用空间对比

3.2 案例研究

地铁作为城市轨道交通的一种，已经成为很多人最主要的交通工具之一，为了增强地铁运行时的安全性，需要对地铁的运营环境有较全面的实时把握，才能在关键时刻做出正确的决断。但由于地铁运行环境的特殊性及无线传感网的便利性，所以多采用无线传感器网检测地铁的运行时数据[16]。以广州地铁为例，其在地铁隧道中部署温度传感器、湿度传感器、速度传感器、电压传感器等等。

以广州地铁为例，列车进站时要考虑的重要因素为车门停靠位置需要对齐屏蔽门，这样才能方便乘客的下车。所以在列车进站前，需要考量关于列运行车及环境的实时数据包括：列车的实时速度，列车的减速加速度及每节车门与对应屏蔽门的距离等等。此外，温度，湿度等这些辅助因素数据也是需要考虑的，在不同的温湿度情况下，导轨的阻力是不同的，进而影响列车的速度，加速度等关键因素。所以如何在复杂的列车运行环境中处理存在依赖关系的实时数据流，实现对列车日常运行状况的实时监控，甚至在列车遇到突发紧急情况时，能快速做出的响应是广州地铁迫切需要解决的问题。

依据基于事件树的事件划分算法的思想如下：

a)根据地铁进站时实际运行环境中传感器节点的种类，结合业务的需求依次赋予传感器检测事件的优先级如表 2 所示。

表 2 地铁检测事件优先级分类表

事件名称	事件 ID	优先级类型	优先级值
加速度	ACCE	主要事件	3
速度	SPEE	主要事件	3
距离	DIST	主要事件	3
温度	TEMP	主要事件	3
湿度	HUMI	主要事件	3
关键加速度	CACC	关键事件	4
关键速度	CSPE	关键事件	4
关键距离	CDIS	关键事件	4
平均加速度	AACC	非关键事件	1
平均速度	ASPE	非关键事件	1
平均距离	ADIS	非关键事件	1
入站	ENTE	正常事件	2
出站	OUTB	正常事件	2

表 2 描述了事件名称、事件 ID、优先级种类和赋予的优先级值。其中主要事件指传感器实时发回的检测事件；关键事件指某一特殊时刻时传感器检测到事件，事件产生的影响十分明确。

b)在事件的有效长度窗口内，参照表 2 的事件优先级构造以复杂事件为根节点的事件树。如图 6 所示（其中 A，B，⋯ Y，Z 为任意值）。

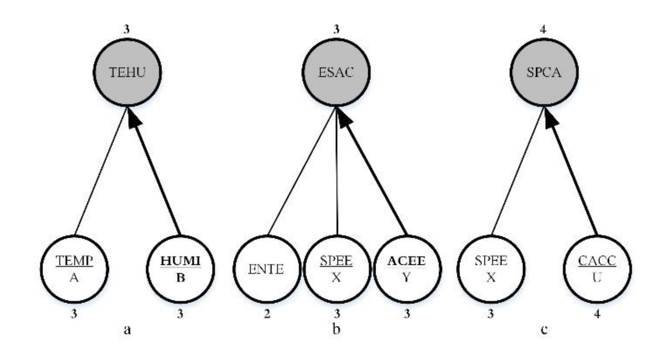


图 6 列车进站时对应的事件树

如图 6 所示，事件树 a 表示环境中此时的温度为 A，湿度为 B，由于其优先级相同，所以将两者的任意优先级值传给根节点；事件树 b 表示列车以速度为 X，减速加速度为 Y 的速度进站，其中速度与加速度的优先级值高，将该值传给根节点；事件树 c 表示速度 X 以关键减速加速度为 U 减速，其中关键加速度具有的优先级最大，将其优先级值传给根节点。

c)由于上图 6 中的事件树 b 与事件树 c 都包含以速度为叶节点的事件树，所以将这两个存在依赖的事件树存放在一个事件树链表中。其中链表中的索引值为每个事件树中根节点的优先级值。

d)在 c)构造的事件树链表的基础上，当任何一个 CEP 引擎请求处理该链表中的事件树的叶子节点事件时，则将该事件树链表里的所有事件推送给该 CEP 引擎处理。如图 7 所示。

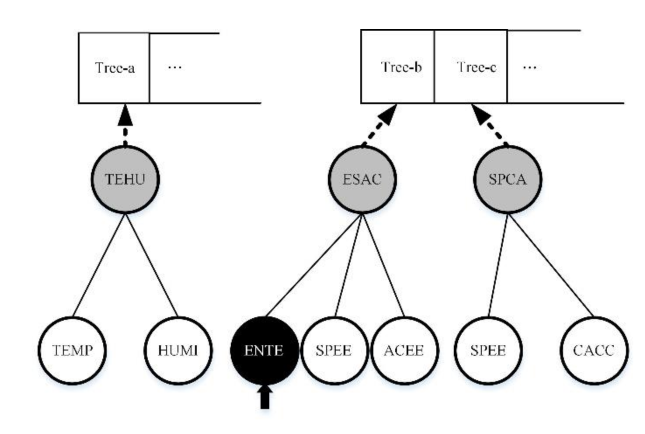


图 7 列车进站时对应的事件树分流

如图 7 所示，当列车进站时，复杂事件检测模块检测到 ENTE 事件被触发，CEP 引擎开始处理以入站事件 ENTE，速度事件 SPEE，加速度事件 ACEE 构成的 b 事件树，以期得到复杂事

件 ESAC；同时 CEP 引擎也要处理速度事件 SPEE 和关键加速度事件 CACC 构成的 c 事件树以得到复杂事件 SPCA，但因为在该时间窗口内同时存在以速度事件 SPEE 相同为叶子节点的事件树 b 与事件树 c，所以只有将事件树 b 和 c 放在同一个 CEP 引擎中处理，才能得到整个过程中产生的复杂事件。

综上所述，该复杂事件数据流最终描述的是地铁列车在温度为 A，湿度为 B 的环境下，原本以 X 速度，减速加速度为 Y 进站，进站后发现速度过高，开始调整关键减速加速度为 U，以适应列车的停靠。

4 结束语

本文着重研究针对实时数据复杂事件检测中的事件负载分流问题，提出了一种基于事件树的多依赖复杂事件检测方法。该方法继承了基于树型结构的复杂事件检测的高效性，并进一步给出异构事件源之间多依赖关系的明确定义，使用一种依赖事件树链的方法对海量原始数据流进行分流处理，提高了复杂事件检测效率和事件吞吐量。案例研究说明了该方法在地铁列车运行环境中的可行性。

针对具体的应用场景，对于该方法在实际应用过程中遇到的新的特点，有待进一步深入研究。

参考文献：

[1] Ma Meng, Wang Ping, Chu C H, *et al.* Efficient multipattern event processing over high-speed train data streams [J]. IEEE Internet of Things Journal, 2017, 2 (4): 295-309.

[2] Giatrakos N, Artikis A, Deligiannakis A, *et al.* Complex event recognition in the big data era [J]. Proceedings of the VLDB Endowment, 2017, 10 (12): 1996-1999.

[3] Mei Yuan, Madden S. ZStream: a cost-based query processor for adaptively detecting composite events [C]// Proc of ACM SIGMOD International Conference on Management of data. New York: ACM Press, 2009: 193-20.

[4] Sun Jinan, Huang Yu, Huang Shuzi, *et al.* Formal method based on Petri nets to detect RFID event [J]. Journal of Computer Research & Development, 2012, 49 (11): 2334-2343.

[5] Liu Hongying, Goto Satao, Li Junhua. The study and application of tree-based RFID complex event detection algorithm [C]// Proc of International Symposium on Web Information Systems and Applications. 2009: 520-524.

[6] Ji Ke, Zhan Yongzhao, Chen Xiaojun, *et al.* Detection of complexity video event based on hypergraph model [J]. Application Research of Computers, 2012, 29 (12): 4770-4774. .

[7] Schultz-Møller N P, Migliavacca M, Pietzuch P. Distributed complex event processing with query rewriting [C]// Proc of the 3rd ACM International DEBS Conference on Distributed Event-Based Systems. New York: ACM Press, 2009: articleNo 4.

[8] Akdere M, Tatbul N. Plan-based complex event detection across distributed sources [J]. Very Large Data Base Endowment, 2008, 1 (1): 66-77.

[9] 陈皓，李瑜，虎嵩林，等. 基于 S4 框架的并行复杂事件处理系统 [J]. 通信学报，2012, 33 (Z1): 165-169. (Chen Hao, Li Yu, Hu Songlin, *et al.* Parallel complex event processing system based on S4 framework [J]. Journal on Communications, 2012, 33 (Z1): 165-169.)

[10] Hedtstück U. Complex event processing [J]. Methoden Und Innovative Anwendungen Der Informatik Und Informationstechnik, 2009, 51 (5): 241-242.

[11] 孟强. 应用 RFID 技术实现地铁列车精确定位的研究 [J]. 科技信息，2010 (33): 77-78. (Meng Qiang. Application of RFID technology to realize accurate positioning of subway trains [J]. Sience & Technology Information, 2010 (33): 77-78.)

[12] Wang Di, Rundensteiner E A, Ellison R T. Active complex event processing over event streams [J]. Proceedings of the VLDB Endowment, 2012, 4 (4): 634-645.

[13] Kolchinsky I, Sharfman I, Schuster A. Lazy evaluation methods for detecting complex events [C]// Proc of the 9th ACM International Conference on Distributed Event-Based Systems. New York: ACM Press, 2015: 34-45.

[14] Xu Chuanfei, Lin Shukuan, Lei Wang, *et al.* Complex event detection in probabilistic stream [C]// Proc of the 12th International Asia-Pacific Web Conference. Washington DC: IEEE Computer Society, 2010: 361-363.

[15] Verssimo P E, Aniello L, Luna G A D, *et al.* Collaborative inter-domain stealthy port scan detection using esper complex event processing [M]. Berlin: Springer, 2012: 139-156.

[16] 利浩能. 浅谈 RFID 技术在地铁行业的应用 [J]. 科技信息，2011 (22): 379-379. (Li Haoneng. Talking application of RFID technology in subway industry [J]. Sience & Technology Information, 2011 (22): 379-379.)